The newpax package, v0.56

Reinserting annotations from included pdf file

Ulrike Fischer*

2025-08-21

1 Introduction

Links in a PDF are created with annotation objects. Such an object is not connected to the content or text, but simply describes an (rectangular) area on the page and defines an action if the cursor is in the area. The coordinates of the area are given in absolute page coordinates. The action of such an annotation can be an external URL, but also an internal destination. Such destination are objects describing a page and some instructions how to display the page—again using absolute coordinates.

When a PDF is included in another PDF—may it be with `\includegraphics` or with `\includepdf`—the annotation coordinates no longer make sense as they don't refer to the receiving page (and often the action of an annotation doesn't make sense either), so all TeX-engines and backends strip them away when including a PDF: the net effect is that external and internal links are lost.

The `pax` package from Heiko Oberdiek offers a solution for this problem: it extracts all the annotations and destinations of the included PDF in a text file, does some clever recalculations of their coordinates and reinserts them. The package works basically fine but has a few drawbacks: To collect the annotation one has to run an external java program which relies on a now outdated library, and it works only with pdfLaTeX.

The `newpax` tries to address these problems. It offers a lua script to extract the annotations. The script can be used with lua(la)tex and no external tools are needed. The annotations can then be reinserted either with the `pax.sty` or with the new `newpax.sty` whose code is based in large parts on the `pax` package: it uses its data structure and the original code to calculate the coordinates (with a few minor bug corrections), but the pdfLaTeX primitives have been replaced by commands from the LaTeX PDF management in `pdfmanagement-testphase` so it should works with all major engines and backends (with the exception of dvips).

*fischer@troubleshooting-tex.de

2 Quick use instructions

2.1 Step 1: extract and collect the annotations

The lua script offers a function which take as argument the name of a PDF (without the extension). The function can be used in some lua scripts but also in a document which then must be compiled with lualatex.

Listing 1: doc-extract-newpax.tex

```
\documentclass{article}
% load the lua code
\directlua{require("newpax")}
% write .newpax files for newpax.sty
\directlua
  {
    newpax.writenewpax("doc-input1")
    newpax.writenewpax("doc-input2")
  }
\begin{document}
\end{document}
```

Running this document will create the files `doc-input1.newpax` and `doc-input2.newpax`.

To find the graphics `kpathsea` is used. This means that graphics in texmf trees will work and you can also use paths to directories, but settings in `\graphicspath` are ignored. The newpax file is currently written into the current directory, which means that graphics with the same name in different locations won't work easily (with lualatex you could create the newpax file in the document just before it is needed). Later versions of the package will probably add some options for this case, but for now use at best distinct file names.

2.2 Step 2: Using the `.newpax`-file with `newpax`

The package `newpax` is based on the package `pax` but extends it in various way. It requires the LaTeX PDF management code in the `pdfmanagement-testphase` package. The PDF management is *not* compatible with every package, check its documentation!

The following listing shows how to use `newpax`.

- It should work with pdflatex, lualatex and xelatex. The latex/dvips route fails as this can't include PDF anyway.
- Some provision have been added to allow multiple inclusion of the same PDF, but if you insert different sets of pages from a PDF some destinations can still be missing. So better avoid it.
- You can choose for every file if border color and styles of links are taken from the source PDF or from the hyperref settings. But you can't adjust or change colored links.

- You can add additional settings to the annotations, for example an `/F` flag, with

```
\ExplSyntaxOn
\pdfannot_dict_put:nnn {link/URI}{F}{4}
\ExplSyntaxOff
```

Listing 2: doc-use-newpax.tex

```
% The next command needs LaTeX 2022-06-01, for older formats see documentation
% of pdfmanagement-testphase
\RequirePackage{pdfmanagement}
\documentclass{article}

\usepackage{pdfpages,xcolor}

\usepackage{hyperref}
\hypersetup{linkbordercolor=blue}

\usepackage{newpax}

%use the link border color and style of the imported pdf
%and not hyperref colors
\newpaxsetup{usefileattributes=true}

\begin{document}

\includegraphics[scale=0.5,trim=4cm 15cm 8cm 3cm,clip,page=1]{doc-input1}
\includegraphics[scale=0.5,trim=5cm 15cm 8cm 3cm,clip,page=2]{doc-input1}

%set a unique suffix if the pdf is imported twice
\newpaxsetup{destsuffix=B}
\includegraphics[scale=0.5,trim=4cm 15cm 8cm 3cm,clip,page=1]{doc-input1}
\includegraphics[scale=0.5,trim=5cm 15cm 8cm 3cm,clip,page=2]{doc-input1}

% suppress the adding of annotations
\newpaxsetup{addannots=false}
\includegraphics[scale=0.5,trim=4cm 15cm 8cm 3cm,clip,page=1]{doc-input1}

%reactivate, don't use file attributes
\newpaxsetup{addannots=true,usefileattributes=false}
\includepdf[pages=-]{doc-input2}
\end{document}
```

2.3 Combining the steps

When using lualatex both step can be simply in the same document. With other engines you can use `ifluatex`.

3 Setup options

```
\newpaxsetup{key-val option list}
```

This command allows to change the behaviour inclusion. It knows the following keys:

`usefileattributes` This is a boolean key. If set to true, the reinserted annotations will use the linkborder settings (color and style) of the included file, if set to false, the settings of the receiving PDF will take precendence.

`destsuffix` This allows to add a suffix to the destination names. This is needed if a file with destinations is included more than once, to avoid to get multiple destinations.

`addannots` This is a boolean key. It allows to switch on and off the reinserting of the annotations. When set to false it also suppress warnings in the log if the `.newpax` file is not found. It is recommended to set it to false for graphics which don't have links.

`dests` This is a choice key. Currently the values `used` (the default) and `all` are allowed. In the first case only destinations that are targets of links in the included PDF will be included, in the second case all destinations (if they are in the included pages) will be included. The second can be useful if you want to link to destinations "from the outside", see below section 6.1.

4 More Background

Clickable links in a PDF are one example of an annotation. Annotations are areas on a page which are associated with an action. A typical annotation object could look like this in the PDF:

```
15 0 obj
<<
/Type /Annot
/Subtype/Link
/Rect [147.716 654.025 301.887 665.15]
/Border[0 0 1]/BS<</S/U/W 1>>/H/I/C[0 1 1]
/A<</Type/Action/S/URI/URI(https://www.latex-project.org)>>
>>
endobj
```

This is an object of type `Annot` and subtype `Link`. The `/Rect` value describes the rectangle of this annotation. The coordinates are absolute coordinates related to the current page. It is important to understand that an annotation is not connected to some page content but only to a location! The `/Border` setting and the other values in this line describe the look and color of annotation. The `/A` value contains the action, in this case it is an url to an external website.

To "reactivate" the annotations of an included pdf one has to do a number of tasks.

- One must *retrieve and store* the annotations of the included pdf. For links to external url's this requires to find only one object like the one shown above. But e.g. internal links point to destination objects and these must be found too.
- One must *recalculate* the rectangle coordinates to fit to the coordinate system of the target page: as the included pdf can be placed at various positions, scaled, rotated and even clipped this is not an easy task. Destinations have rectangles too that must be recalculated.
- One must *reinsert* the annotation and related objects. This has to take into account that a pdf is perhaps not included completely, a link shouldn't point to a missing page or a clipped annotation. It also has to take into account that a pdf is perhaps inserted more than once or in steps.

4.1 Retrieving and storing annotations

Theoretically one can do it manually: Uncompress the PDF (or when using LaTeX, create directly an uncompressed one), open it in an editor and copy and paste all needed objects. Practically one naturally want some tool.

The `pax` package from Heiko Oberdiek consists of a perl script and a java-jar file `PDFAnnotExtractor` which can extract the necessary objects. It writes the information to a file with the extension `pax`. When it has been successfully installed it works quite fine. Problems with this approach are

- `PDFAnnotExtractor` requires an external, old version of the java library of PDFbox which must be installed manually;
- it requires a java installation and
- it is not extensible.

The `newpax` package comes with a lua-file. It uses the `pdfe` library embedded in luatex to extract the annotations and other needed information. `newpax` writes the information to a file with the extension `pax` or `newpax`. The content of the files is (nearly) identical to the content of the `pax`-file written by `PDFAnnotExtractor`. The lua code was written by looking at example outputs from `PDFAnnotExtractor` and reproducing it in lua. The ordering of some elements is a bit different and some strings are output in a different way but for the examples I used the resulting `pax`-files can be used together with the original `pax.sty`. But due to the fact that the code was written without real spec simply by looking at examples, it is quite probably that the lua code is not yet handling all objects or options that `PDFAnnotExtractor` outputs. But the code can rather easily be extended when the needs arises.

The code also doesn't handle structure elements, neither at the export nor at the import. I have yet no real idea what would be sensible here (and I'm quite sure that `PDFAnnotExtractor` doesn't handle this either.)

5 Importing annotations

The import function has to handle two main problems:

- Recalculation of coordinates if the imported PDF is scaled or moved
- Dropping of annotations and destinations if a PDF is only partially included, e.g. because the graphic is clipped, or because only a selection of pages are included.

The `pax` package from Heiko Oberdiek does here hard work to recalculate the annotation rectangles and to decide which annotation and which destination should be reinserted and patches `\includegraphics` command to automate this. `newpax` mostly reuses the core commands of `pax` and added only a number of switches and support for more engines and backends.

6 Internal links

Internal links (GoTo links in PDF speach) are more complicated than the other link types.

At first they involve two objects: the link annotation and the target of the link (destination in PDF speach). If a PDF is included partially, it is therefore not enough to check if the link area is on the visible pages, one also has to check if the target of the link is there. As such a target can be on a later page the `pax`/`newpax` uses the `.aux` file to record which targets exists and which are required and in a second compilation decides which links and destinations should be reinserted. As an example: if you include from a PDF the table of contents (with links) and a few pages, only the links in the toc pointing to visible sections *and* only the destinations needed for this links will be reinserted.

At second the targets are normally „named destinations“, that means a link annotation points to a string like `section.1` and the name tree `/Dests` contains a mapping for this string to an destination object. As names like `section.1` are used in many PDFs produced by LaTeX they can not simply be reused when reinserting annotations. `newpax` tries to avoid name clashes by generating names consisting of a prefix with the file name and number or name. So e.g. the `section.1` destination would be called `file.newpax@section.1` in the receiving file. If you set up a suffix, e.g. with `destsuffix=A`, it is appended with an @ symbol, so the result would be `file.newpax@section.1@A`. If the original destination has not name (this can happen if you include PDFs which haven't been created by TeX), then a number is used, so you get `file.newpax@1`.

6.1 Access from the „outside“

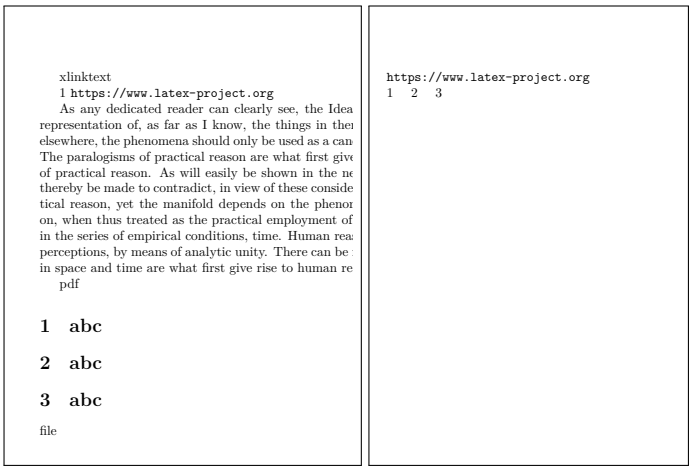
It is possible to link from the external document to destinations in the included PDF, for example to build a table of contents with links. For this you need

- the included PDF must contain all destinations that you want to use. That means, `newpax` won't create destinations out of the blue.

- You must ensure that the destinations you need are imported. This will be the case if they are targets of internal links in the imported PDF, if not you can force that all destinations of visible pages are there by using the option `dests=all`, see above.
- You need the names of the destinations. If the imported PDF has been created with LaTeX you can look in the `toc` or `aux`-file to find names. Then you can setup a filter to link to the names used by `newpax` and e.g. load the toc or copy some of the content lines:

```
\newpaxsetup{dests=all}
\def\HyperDestNameFilter#1{myinput.newpax@#1}
\input{myinput.toc} %load toc
\includepdf[pages={2-4}]{myinput}
```

7 Example input



Check also the output of the listing above, `doc-use-newpax.pdf`.

8 Support for the `pax` package

8.1 Step 1: Extracting the annotations

The lua script is also able to write `pax` files for the `pax` package (and so can be used to replace the java application).

For this extract the annotations like this:

Listing 3: doc-extract-pax.tex

```
\documentclass{article}
% load the lua code
\directlua{require("newpax")}
```

```
% and/or write .pax files for pax.sty
\directlua
  {
    newpax.writepax("doc-input")
    newpax.writepax("doc-input2")
  }
\begin{document}
\end{document}
```

8.2 Step 2: Using the `.pax`-file with `pax.sty`

Ensure that the `.pax` file created in step 1 can be found by your main document. You can then insert your PDF files together with their annotations like in the following listing.

- This works with pdflatex and lualatex. lualatex needs the extra code demonstrated in the document.
- It needs two or three compilations until every reference is correct.
- There is a small typo in `pax.sty` which affects clipping, the patch shown in the listing correct this.
- In some cases the catcode of # and % must be set to letter to avoid errors.
- Don't include PDFs with destinations twice as this will lead to duplicate destinations and pdflatex will complain.
- If annotations should not be reinserted remove the `.pax`-file.
- If `hyperref` is loaded you can change the color and style of link borders with hyperref options.

Listing 4: doc-use-pax.tex

```
\documentclass{article}
\usepackage{ifluatex,etoolbox}
\usepackage{pdfpages}
%pax needs this to run with lualatex
\ifluatex
\usepackage{pdftexcmds}
\makeatletter
\let\pdfstrcmp\pdf@strcmp
\let\pdfescapename\pdf@escapename
\makeatother
\usepackage{luatex85}
\fi
%load pax
\usepackage{pax}
%correct a bug in pax affecting clipping
\makeatletter
```

```
\patchcmd\PAX@pdf@annot{\PAX@pagellx}{\PAX@page@llx}{}{\fail}
%allow hashes and percent in the pax file
\patchcmd\PAX@AddAnnots{\InputIfFileExists\PAX@file{}{\typeout{* Missing: \PAX@file}}}
 {\begingroup \catcode`\#=12 \catcode`\%=12
  \InputIfFileExists\PAX@file{}{\typeout{* Missing: \PAX@file}}\endgroup}{}{\fail}
\makeatother
\begin{document}
\includegraphics[scale=0.5,trim=5cm 15cm 8cm 3cm,clip,page=2]{doc-input1}
\includegraphics[scale=0.5,trim=5cm 15cm 8cm 3cm,clip,page=1]{doc-input1}

\includepdf[pages=-]{doc-input2}
\end{document}
```